Interactive Distance Lab Solution for Robotic Arm

Kadri Umbleja[1], Martin Jaanus[2], Vello Kukk[3] and Andres Udal[4]

[1] Tallinn University of Technology, Tallinn, Estonia, kadri.umbleja@ttu.ee
[2] Tallinn University of Technology, Tallinn, Estonia, martinj@ttu.ee
[3] Tallinn University of Technology, Tallinn, Estonia, vello.kukk@dcc.ttu.ee
[4] Tallinn University of Technology, Tallinn, Estonia, andres.udal@ttu.ee

Abstract

Teaching industrial robotic arms has been hold back by expensive equipment. Virtual laboratories offer cheaper alternative for full-scale labs but they lack the hands-on experience and real, unpredictable, results. Another alternative is distance lab that offers student ability to use real equipment from the distance. Also, it offers safety for robotic arm experiments as unexpected movements cannot harm the learner. In this paper, distance lab solution for small scale robotic arm similar to ones used in the industry is introduced. It uses novel time-sharing algorithm that allows multiply student to work with the same robot on the same time. The laboratory is accessible 24/7, uses automatic power and lights control. To make the learning more interactive, controllable LED board was developed. The developed solution is fully web based system that only requires web browser for access. Robot's safety is considered. Also, working with object is solved, including situation when more than one student is accessing the robot and objects have been picked up. Novel evolutionary algorithm approach was used for inverse kinematics solution.

Keywords: *Robotics, Distance laboratory, Competence-based learning, Distant learning, Evolutionary algorithm, Inverse kinematics, E-learning, Time-sharing, Resource sharing.*

1. Introduction

The industrial robotic arms usually use high pneumatic and/or electrical power. Also, real industrial robotic arms are expensive. Furthermore, during the learning process students may not have enough knowledge to operate the equipment safely and might harm themselves or the machine. Therefore, on-site classical labs may not be the best solution and may not be even an option due to the financial circumstances as multiple copies of the same equipment is needed.

Learning about controlling and using industrial robotic arms is crucial knowledge in the field of automatization and computer control. Therefore, alternative solutions for not feasible classical on-site labs have to be considered. Following options are possible: on-site simulation, distant usage of the equipment and virtual laboratory [1].

In this paper we consider distant laboratory option. We replaced expensive industrial arm with smaller downscaled robotic arm developed specially for teaching industrial robotics [2]. Our arguments were that as we want to utilize the expensive resource as much as possible, distant lab will not enforce any location or time restrictions for learners like classical lab solution. Also, instead using virtual laboratory, using real equipment will keep the uncertainty of the experiments – unexpected results can occur that are lacking in simulations. That would guarantee that learning process resembles real life situations and problems that might occur [3].
Therefore, distant laboratory solution was developed.

2. Robolab

The concept behind Robolab is to integrate robotic arm lab experiment into department's novel competence based e-learning environment [4]. Laboratory equipment would be accessible for the learners the same way inside and outside the university - learner only needs web browser with internet connection. They would be able to work in time and place of their own choosing. Students have access to camera feeds where they can follow the robotic arm, situated within university. They follow the instructions in the e-learning environment to complete the experiments, submit their answer and observe how their solution works live using the cameras. Automatic evaluation is used to assess the result.

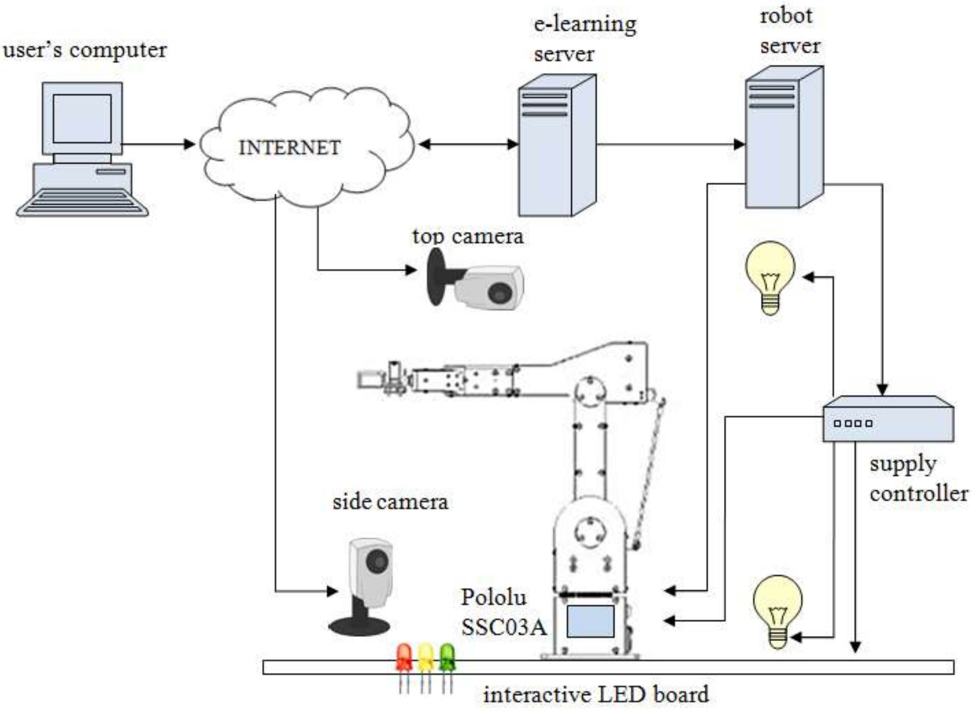

Figure 1. The design of robolab.

Robolab consists of robotic arm, server, two cameras and supply controller as shown on Figure 1. The basic behaviour and movement of the robot arm is the same as in the case of real the industrial robot arms. "Erik", the robotic arm used in the Robolab, is specially produced for educational purposes by Robotnik Automation S.L.L. [5] as seen on Figure 2. It has six degrees of freedoms plus the gripper function as in the case of the typical industrial robot arm. The robotic arm uses the standard servo motors that are interfaced with "Micro Serial Servo Controller Pololu SSC03A" [6] equipped with RS232 port.

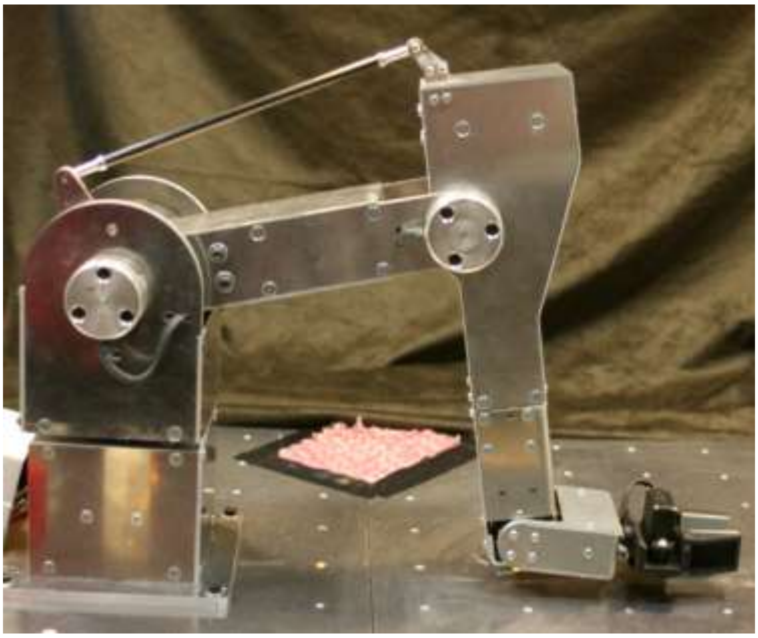

Figure 2. Robotic arm “ERIK”, used in Robolab.

Robot is controlled by its own dedicated server that contains software running as a web application. E-learning environment uses AJAX requests to send commands to robot’s server software. Software at robots server can communicate directly with micro controller in robotic arm over COM-port. The software takes high-level commands and on robot’s server side converts them into input sequences that microcontroller can understand.

Also, the software on robot’s server sends commands over COM-port to supply controller. Supply controller is used to turn electricity on and off. It controls power to the actual robotic hand, to the lights in robot’s surroundings and multiply LED lights used for marking coordinates and axes on robot’s board. Power supply contains simple Atmega88 8-bit microcontroller that uses auxiliary chip FT232RL at input side to connect microcontroller with robot’s server. On output side, the microcontroller has two auxiliary chips ULN2003 that are the 7-channel Darlington transistor buffers to assure the turn-on of the necessary 220 VAC power relays and the LED board lights.

Two cameras, one at the side and other at the top of the robot, are used for monitoring the experiments. At the end of every movement made by robot, screencaps from both cameras are saved for the logging purpose. Student can access his/her own log with servo positions and images. Log also allows to rerun the entry.

3. Time Sharing

One of the main aims of Robolab concept was that it is accessible for as many people as possible, including learners outside university. The aim, also was to encourage learning, popularize robotics and give students possibility to do experiments that were not available for them before.

As there is only one robotic arm due to the high cost of the hardware and the possibility of many learners some kind of resource sharing method had to be developed in order to achieve our goals.

The first and most simple solution would have been that learner books the robotic arm for some specific time and has monopoly access to the equipment. That solution has many disadvantages. First, learning process contains of different stages and using the equipment to run the experiments takes actually only a small portion of the overall learning time. If monopoly access is used no other student than one who currently has booked the robot could access it, even though no experiment is running. Also, there might be the case when robot is booked but student forgets the booking or his/her plans changes – robot will stay unused.

To overcome those drawbacks and achieve set goals, time sharing system with no booking was developed. Instead of sending commands directly to the robot, they are saved to queue in database and server acts as a middle-man forwarding them to the robot following queue management. First come, first served principle is used for majority of the cases with few exceptions described later. That allows students to run their experiments whenever they want at all times and multiple students can work in the same time.

That approach requires some additions. First of all, as camera feed can be followed by everyone all the time, learner has to be able to be able visually distinguish when his/her experiment is running. That is implemented with the background color of camera windows. When the experiment, currently followed, belongs to that student, background is changed to yellow to draw the attention.

Also, countdown clock is used to inform student how long it will take before his/her experiments starts to be executed. In order to properly predict time for execution, an algorithm that calculates how long the command will take to execute according to the robot's previous position and servo movement speeds had to be developed. As robot's movement depends on its previous position, it is important for continuous workflow that robot will take the position robot had when finishing user's previous command before starting to execute new command – that will cause small delay between commands.

There is exception when first come, first served principle is not followed strictly. The special case is when learner is working with objects. If robot has picked up object from the table, that user has monopoly access over the robot for next three minutes. If during that time, the learner gives new command, that time is reset after every movement until object is put safely on the board. If learner does not continue working the object is automatically returned to the board via pre-programmed sequence and resource is freed for shared usage.

Time sharing system allows students to work on time of their own choosing with minimal waiting and interruption.

4. Safety

As distant laboratory solution is used, learner is separated from the equipment and cannot hurt himself/herself with unexpected movement of the arm. Therefore, we only needed to consider the safety of the robotic arm.

All commands, sent to the server, have to go through safety check. It is verified that those commands do not hurt robotic arm meaning that during the movement none of robot's joints tries to achieve z coordinate below 0 (level of the board). Also, it will check that robot will not strike against itself as could be seen on Figure 3.

To check those constraints, simulation based on D-H equations is used [7]. According to this methodology, the 4x4 matrices are formed that contain information about each joint length, position and angle. Four matrices (one rotation, three translations) are multiplied to find coordinate frame for every joint [7]-[9]. As every joint has its own coordinate frame, gripper and servo coordinates have to be converted into base coordinate frame. It is done by multiplying coordinate frame matrices with each other to form a chain of conversion. Then, matrix with position information is also multiplied with that conversion matrix [8] [9].

As it is one of the most common calculations done in the system and it is required before every command, pre-processing time is optimized by pre-calculating conversion matrix once. Furthermore, instead of using matrix multiplication, system of equations giving x, y, z coordinates depending on angles between joints is extracted from the conversion matrix by working out close form solution. That gives us fast forward kinematic solution.

Figure 3. Robot in hazardous position, pushing against itself (picture from log files).

5. Working with Objects

Lot of learning scenarios with that robotic arm contain working with an object:

a) robot has to pick up blocks from the board in front of it and put them in some other location;
b) robot has to push some button or switch on touch-controlled light;
c) robot has to build something with lego bricks it can access.

In all of those cases it is known where the object is on the board in front of the robot. To complete the task, learner has to move the robot over the object and perform some kind of action. It is complicated to calculate how to move the robot hand so that object would be within robot's reach. Also, automated scenarios that pick up objects from active learning area have the same problem. These are problems of inverse kinematics which enables to derived angles from desired end coordinates.

Inverse kinematics is more complicated process than forward kinematics used for robot's safety. There are many different algorithms to solve that problem. For example: analytical approach, interactive joint movement, algebraic solution, cyclic coordinate descent (CCD) method and Jacobian matrices [8] [10]-[12]. Furthermore there are possibilities that there are no solutions or that multiple possible solutions to achieve the same coordinates exist. Handling those situations had to be considered.

Evolutionary algorithm was chosen to solve inverse kinematic problem. Analytical and geometrical approaches would have been preferred but due to robotic arm's configuration, those solutions gave time consuming non-linear set of equations that were not feasible due to our goal to minimize time between different commands. CCD and Jacobian methods in our implementation took more time.

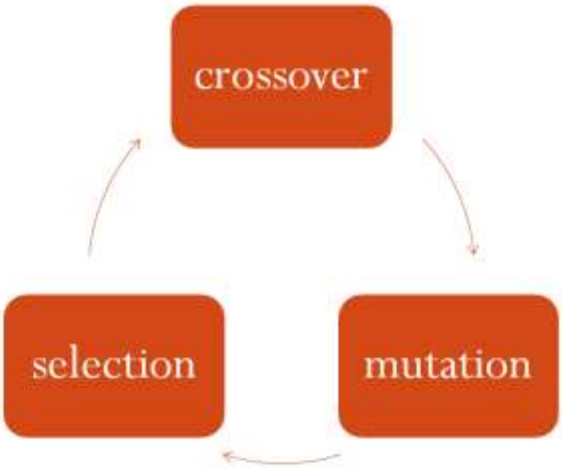

Figure 4. Iterative process of evolutional algorithm.

Evolutionary algorithm uses simple principles from nature to find best possible set of angles for robot's six joints. Evolutionary algorithm uses sets of possible answers (individuals) and applies cross-over, mutation and selection iteratively to find optimal solution as seen on Figure 4.

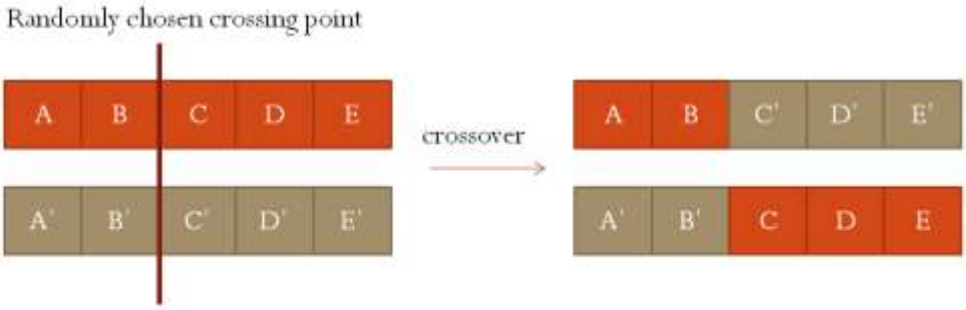

Figure 5. Crossover operation.

During cross-over two individuals are cut in randomly chosen breakpoint and swamp the tail parts over to produce two new full length individuals as shown in Figure 5. During mutation part of the population of individuals are exposed to small change. For robot's angles, a random angle is found and randomly changed by one degree as seen in Figure 6. Then fitness function is used to calculate quality of that individual. In current case, forward kinematic is used as it finds coordinates that would be achieved with that set of angles. Then, distance between goal's coordinates and current coordinates is found. Only best individuals (distant between desired solution and current solution is minimal) are kept for next iteration. [13]

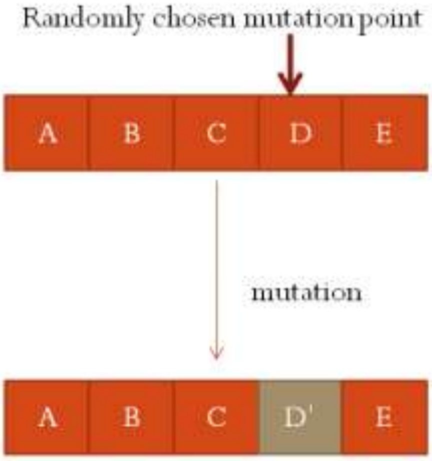

Figure 6. Mutation operation.

That process is continued until a good enough solution is found (we defined “good enough” as Hamilton distance between desired coordinates is no more than 4 mm) or certain number of iterations has been completed (100 is used by default). If solution is found, algorithm returns set of angles. If no solution is found, algorithm will be re-run. No case has been logged when algorithm had to be re-run more than once so far.

The implementation of evolutionary algorithm is very simple and it runs much faster than any other implementations tested. Therefore, it was chosen to be used in the system. Also, it is even further optimized as we left out first joint that is used for robot’s orientation (that could be solved easily with geometry). Also, last join, gripper, was not needed for optimization.

Working with objects adds another complication for safety checks: any movement done by any learner should not change objects positions on active working area. Special area outside the main working area is allocated to store the objects and routines to pick objects up were developed. Also, when there is any object on the active working area, software modifies movements so that the orientation is achieved above board. That would require moving robotic arm higher above the ground without touching any other object, then setting the orientation and moving down to required position, without touching any objects, again.

6. Interactive LED board

In order to make following robot’s movements with camera better to follow and make tasks more interactive, special board with interactively controlled LED lights was designed as seen on figure 7. The green corner lights are reserved for troubleshooting to indicate if power is turned on. Yellow lights form the rulers with 50 mm step to mark the x and y- axes on the ground surface. Red LEDs indicate the special points that are used in exercises where students have to reach certain locations by the robot’s arm or put objects down on the selected coordinates.

Figure 7. Interactive LED board.

7. "Robolab" as learning tool

Robotic arm and its exercises are integrated into competence based e-learning environment and are solved there as any other exercise. When student uses sliders to control the robot, high-level commands are sent to robot's server where they are translated into robot's micro controller commands and transferred to robotic arm. Live feed of robot movement can be followed. Control script that evaluates the correctness of the answer is on the e-learning environment side. Control script can request additional information from robot's server including positions and angels of robot's servos.

Exercises developed for solving with robotic arm can be divided into following themes:
- Simple exercises to familiarize learner with robot, its movement, speed and joints;
- Reading coordinates (indicated by LED lights) from robot's board;
- Simple movement of robot joints to find specific coordinates on robot's board;
- Transforming angles to robot commands and contrariwise;
- Denavit-Hartenberg calculations;
- Running the pre-programmed scenarios and interpreting of the results;
- Writing own scenarios (programming), for example picking up objects.

All the exercises have been divided into different difficulty level so it is guaranteed that student starts with easier tasks before continuing to more theoretical and complex exercises. Majority of the tasks are short, requiring around 10 minutes for solving. Complex programming tasks take longer. The idea of breaking tasks down to smaller portions that do not take too long is to keep student concentrated and focused.

The feedback on the mathematical exercises have been diverse. Some students are glad that they finally have real problem where they can use theory they have learned in other courses and so far had no relevance for them (that has been especially mentioned in context of matrix multiplications). Other students have found those tasks boring and non-interactive compared to the rest of the exercises where robot's movement is at the centre of the task. Students seem to like if they can see their work visually (task ending with robot moving) or if it contains some nice interaction with the board or the environment around robot (tasks with LED lights and programming tasks picking up objects). Students also tend to like shorter exercises not taking more than few minutes. They have found programming exercises interesting and challenging but they are complaining it is time consuming and consider those tasks harder than others (even harder than mathematical exercises).

8. Conclusion

Feedback gathered from the students has shown that they appreciate most the chance to use theoretical mathematical skills acquired in other courses in real practical situations. Also, students are really satisfied with freedom to choose their own time and place of working, not being bound to timetable. Cameras with logs have proven to be enough for students to follow the experiments, to solve the tasks and to draw conclusions.

Amortization of robot has been a problem. Robot has required more maintenance than expected. Also, even with strict safety checks, some of the servo motors had to be changed due to the wear caused by constant usage.

Therefore, it could be concluded that choosing distant laboratory solution for teaching industrial robot control has been the right choice. Time sharing method chosen has proved to be effective and most flexible, utilizing the expensive resource most effective way.

9. Acknowledgements

This work has been partly supported by the Estonian Science Foundation Project 9463.

References

[1] M. Auer, A. Pester, D. Ursutiu and C. Samoila, “Distributed virtual and remote labs in engineering” Industrial Technology, 2003 IEEE International Conference, Vol. 2, pp. 1208-1213, December 2003.

[2] A. Altintas, “A New Approach to 3–Axis Cylindrical and Cartesian Type Robot Manipulators in Mechatronics Education”, Electronics and Electrical Engineering, Vol. 106, No. 10, pp. 151–154, 2010.

[3] D. Feisel and A. J. Rossa, "The Role of the Laboratory in Undergraduate Engineering Education", Journal of Engineering Education, Vol.94, No.1, 2005, pp. 121-130.

[4] K. Umbleja, V. Kukk and M. Jaanus, “Competence-Based Approach to Learning”, *Proceedings of 2013 IEEE Global Engineering Education Conference*, EDUCON 2013, Berlin, Germany, 2013.

[5] “*Homepage of Robotnik Automation S.L.L.*”, http://robotnik.es/en/, Last accessed 27 February 2014

[6] “*Homepage of Pololu Corporation*”, http://www.pololu.com/product/207, Last accessed 27 February 2014

[7] J. Denavit and R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices". Trans. ASME J. Appl. Mech., vol. 23, pp. 215–221, 1955.

[8] R.Jazar, *Theory of Applied Robotics: Kinematics, Dynamics, and Contro*”, New York, Springer Science+Buisness Media, 2007.

[9] R. Manseur, *Robot Modeling and Kinematics*, Boston, Massachusetts: Da Vinci Engneering Press, 2006

[10] J. Craig, *Introduction to Robotics: Mechanics and control,* Upper Saddle River, NY : Pearson Education International, 2005.

[11] J. Latombe, “*Inverse Kinematics*”, http://www.stanford.edu/class/cs273/scribing/scribe4.pdf, Last accessed 27 February 2014

[12] A. Ghosal, *Robotics: Fundamental Concepts and Analysis*. New Delhi : Oxford University Press, 2006.

[13] S. Niermann, “Optimizing the Ordering of tables With Evolutionary Computation”. The American Statistican, Vol. 59, No.1, 2005

Authors

Principal Author: Kadri Umbleja is second year Computer Science PhD student in Tallinn University of Technology. She holds BSc and MSc degree in Computer Science from Tallinn University of Technology.

Co-author: Martin Jaanus holds a PhD degree in Computer Control from Tallinn University of Technology. He is currently an associate professor in Circuit Theory and Design at Tallinn University of Technology. He is Member of IEEE.

Co-author: Vello Kukk holds PhD in Electrical Engineering from Estonian Academy of Sciences. He is currently a professor in Circuit Theory and Design at Tallinn University of Technology. He is IEEE Senior Member.

Co-author: Andres Udal holds PhD in Electrical Engineering from Tallinn University of Technology. He is currently a Senior Research Scientist in Laboratory for Proactive Technologies at Tallinn University of Technology. He is Member of IEEE.